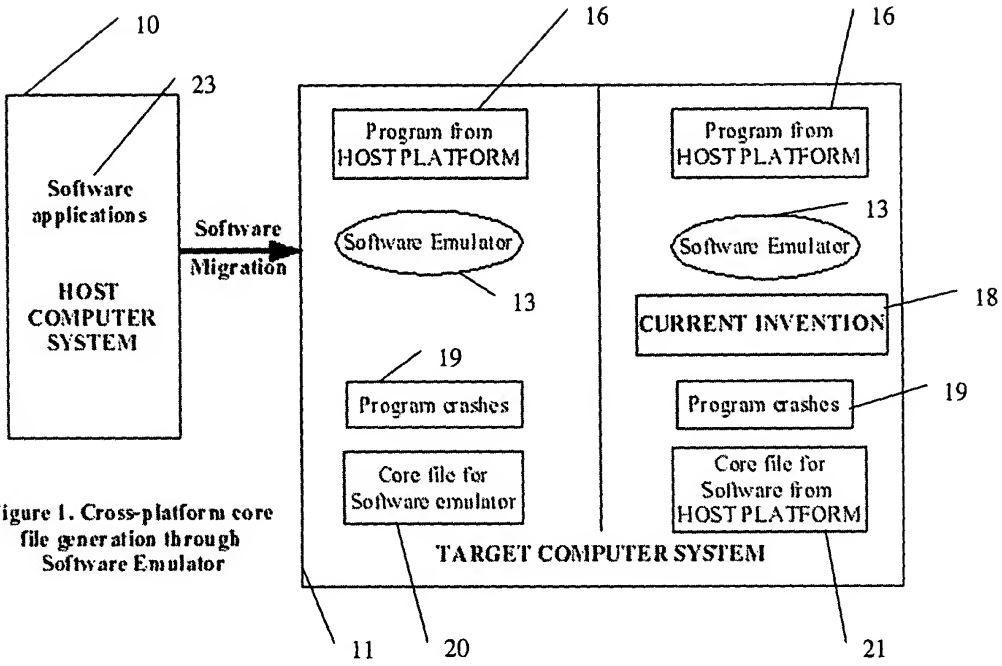
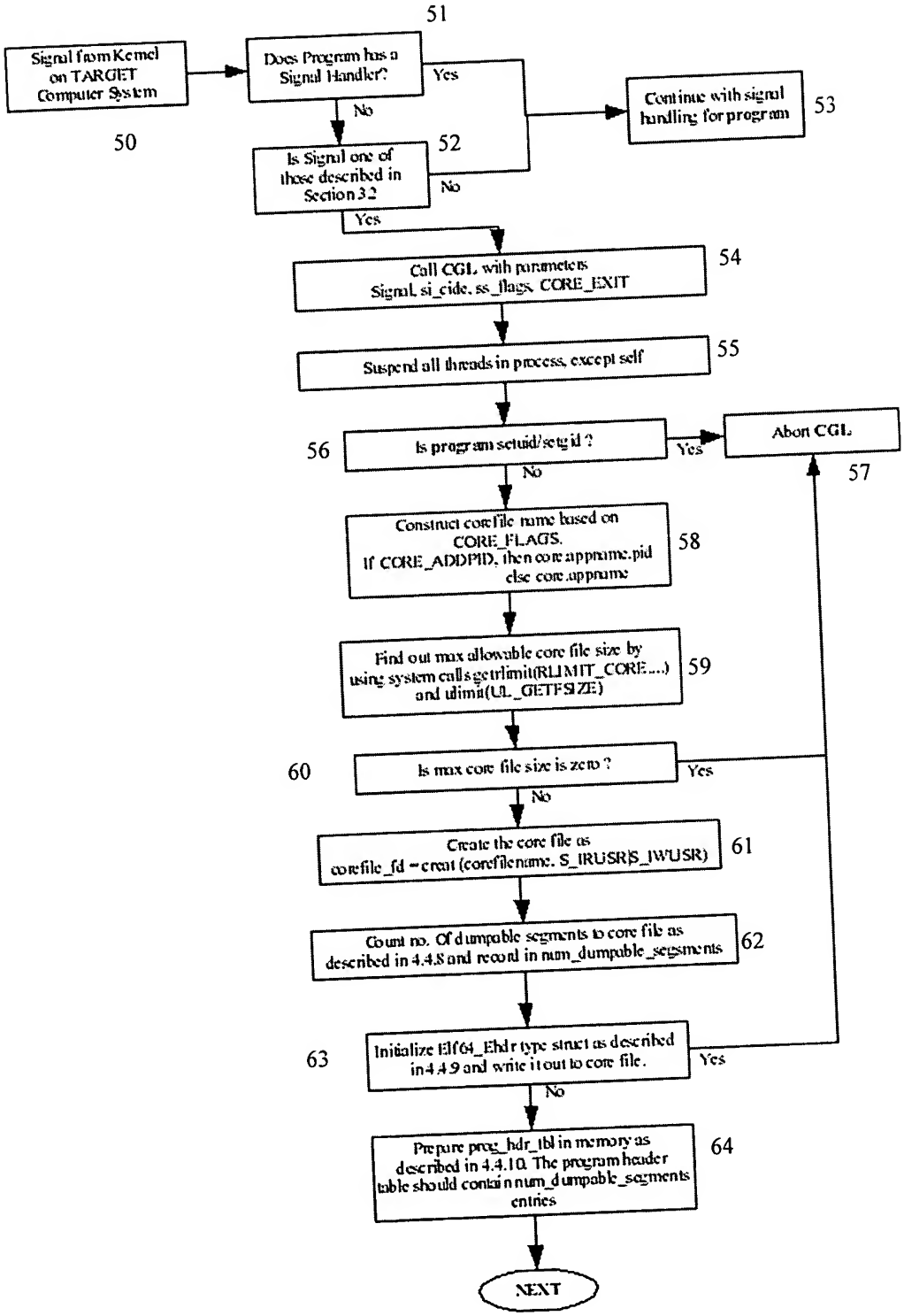



**Figure 1. Cross-platform core file generation through Software Emulator**

| ELF Header |
|---|
| Program Header Table |
| Core Version |
| Kernel Version |
| Command Name |
| proc_info for first thread |
| ..... |
| proc_info for last thread |
| DATA |
| STACK |
| Shared memory (only writable) |
| ..... |
| Shared memory (only writable) |
| Private MMAP Area (Only writable) |
| ..... |
| Private MMAP Area (Only writable) |

**Figure 2. Core file layout for HOST Computer System**


Signal from Kernel on TARGET Computer System
50
Does Program has a Signal Handler?
51
Yes
No
Continue with signal handling for program
53
Is Signal one of those described in Section 3.2
52
No
Yes
Call CGL with parameters Signal, si_cide, ss_flags, CORE_EXIT
54
Suspend all threads in process, except self
55
Is program setuid/setgid ?
56
Yes
Abort CGL
57
No
Construct corefile name based on CORE_FLAGS.
If CORE_ADDPID, then core.appname.pid
else core.appname
58
Find out max allowable core file size by using system calls getrlimit(RLIMIT_CORE,...) and ulimit(UL_GETFSIZE)
59
Is max core file size is zero ?
60
Yes
No
Create the core file as corefile_fd = creat (corefilename, S_IRUSR|S_IWUSR)
61
Count no. Of dumpable segments to core file as described in 4.4.8 and record in num_dumpable_segments
62
Initialize Elf64_Ehdr type struct as described in 4.4.9 and write it out to core file.
63
Yes
No
Prepare prog_hdr_tbl in memory as described in 4.4.10. The program header table should contain num_dumpable_segments entries
64
NEXT

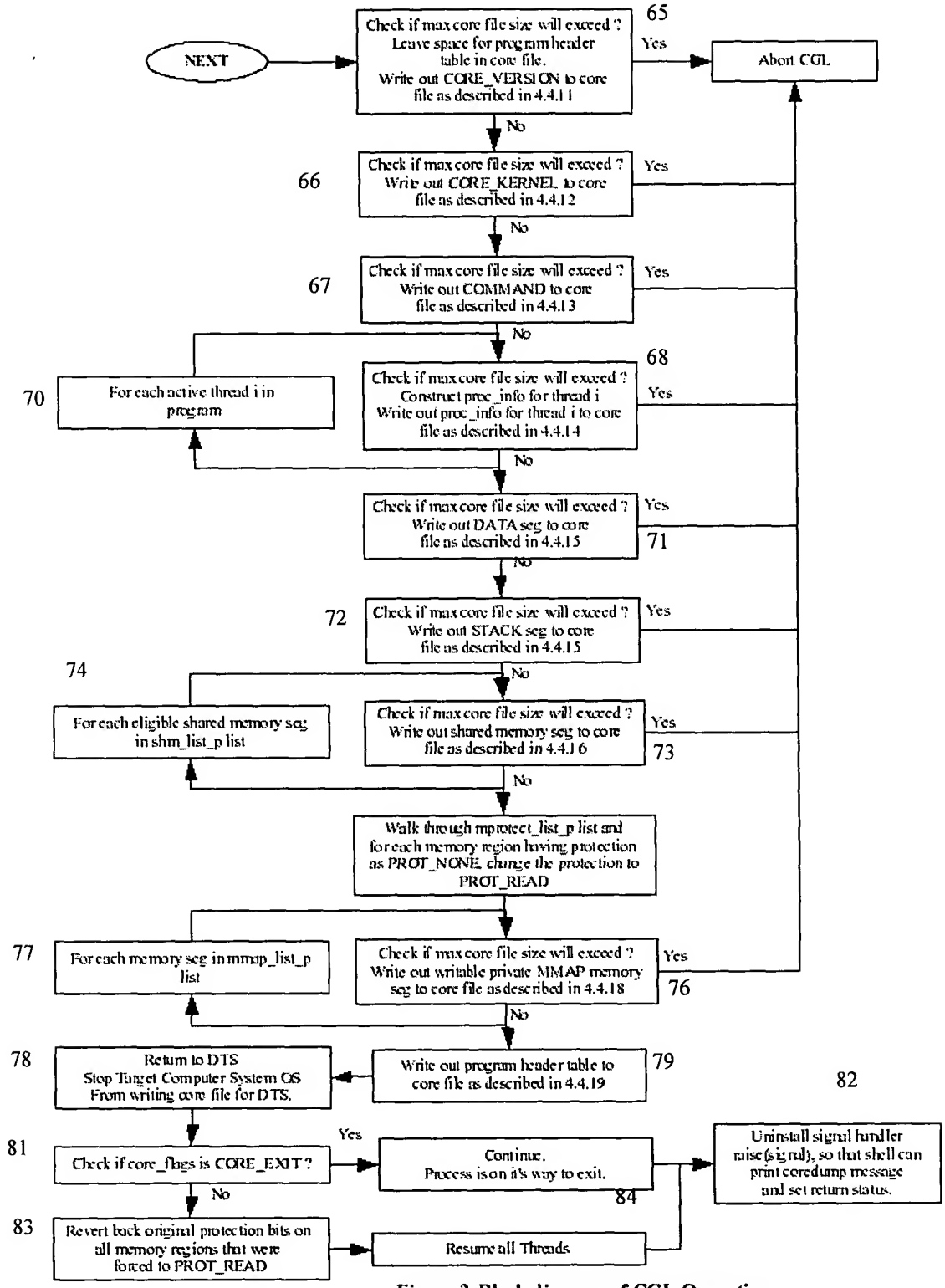



**Figure 3. Block diagram of CGL Operation**